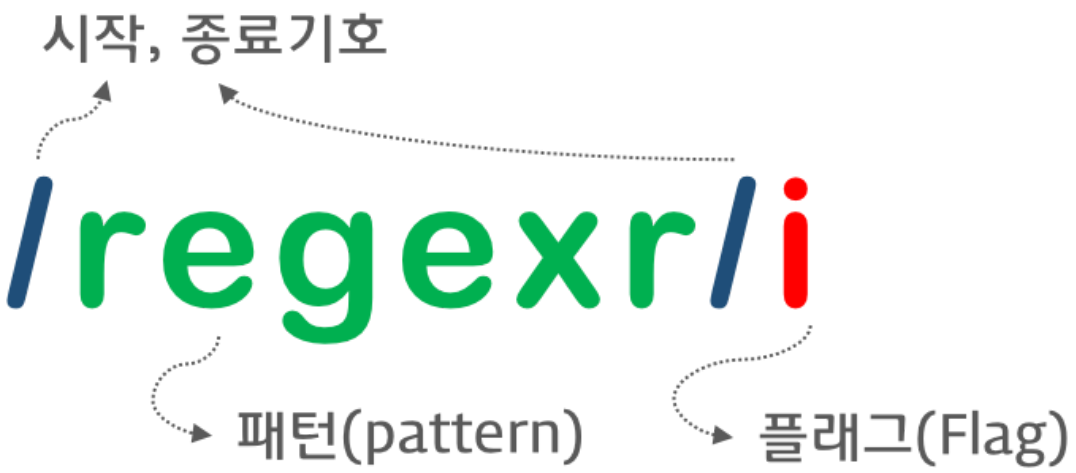
# 31장 RegExp

## 31.1 정규 표현식이란?

- 자바스크립트의 고유 문법 X
- 대부분의 프로그래밍 언어와 코드 에디터에 내장 O

## 31.2 정규 표현식의 생성




- 정규 표현식 리터럴은 패턴과 플래그로 구성

## 31.3 RegExp 메서드

### 31.3.1 RegExp.prototype.exec

- exec 메서드는 인수로 전달받은 문자열에 대해 정규 표현식의 패턴을 검색하여 매칭 결과를 배열로 반환한다.
- 매칭 결과가 없는 경우 null을 반환

```
const target = `Is this all there is?`;
cosnt regExp = /is/;
```

```
regExp.exec(target); // ["is", index: 5,input: "Is this
....]
```

- 문자열 내의 모든 패턴을 검색하는 `g플래그` 를 지정해도 첫 번째 매칭 결과만 반환하므로 주의해야 한다.

---

### 31.3.2 RegExp.prototype.test

- test 메서드는 인수로 전달받은 문자열에 대해 정규 표현식의 패턴을 검색하여 매칭결과를 불리언 값으로 반환한다.

```
const target = `Is this all there is?`;
cosnt regExp = /is/;

regExp.test(target); // true
```

---

### 31.3.2 RegExp.prototype.match

- match 메서드는 인수로 전달받은 문자열과 인수로 전달받은 정규 표현식과의 매칭 결과를 배열로 반환한다.

```
const target = `Is this all there is?`;
cosnt regExp = /is/;

target.match(regExp); //["is", index: 5,input: "Is this
....]
```

- exec메서드와 달리 `g플래그` 를 지정하면 모든 매칭 결과를 배열로 반환한다.

---

## 31.4 플래그

플래그는 총 6개가 있으며 그중 중요한 3개의 플래그를 살펴보도록 한다.

| 플래그 | 의미 | 설명 |
|---|---|---|
| i | ignore | 대소문자를 구별하지 않고 패턴을 검색한다. |
| g | Global | 대상 문자열 내에서 패턴과 일치하는 모든 문자열을 전역 검색한다. |
| m | Multi line | 문자열의 행이 바뀌더라도 패턴 검색을 계속한다. |

- 플래그는 옵션이므로 선택적으로 사용할 수 있으며, 순서와 상관없이 하나 이상의 플래그를 동시에 설정 할 수있다.

---

## 31.5 패턴

- 패턴은 `/` 로 열고 닫으며 문자열의 따옴표는 생략한다.
- 따옴표를 포함하면 따옴표까지도 패턴에 포함되어 검색된다.
- 어떤 문자열 내에 패턴과 일치하는 문자열이 존재할때 '정규 표현식과 매치한다'고 표현한다.

---

### 31.5.1 문자열 검색

```
const target = `Is this all there is?`;
cosnt regExp = /is/ig;

// 플래그 i를 추가하여 대소문자를 구별하지 않고 검색
// 플래그 g를 추가하여 모든 문자열 전역 검색
target.match(regExp); // ["Is", "is", "is"]
```

---

### 31.5.2 임의의 문자열 검색

- `.` 은 임의의 문자 한 개를 의미한다.
- `.` 을 3개 연속하여 패턴을 생성하면 내용과 상관없이 3자열과 매치한다.

```
const target = `Is this all there is?`;
cosnt regExp = /.../g;

target.match(regExp); // ["Is ", "thi", "s a", "ll ", "th
e", "re ", "is?"]
```

---

### 31.5.3 반복 검색

- `{m,n}` 은 앞선 패턴이 최소 `m` 번, 최대 `n` 번 반복되는 문자열을 의미한다.
- 콤마(,) 뒤에 공백이 있으면 정상 동작하지 않으니 주의해야한다.
- `{n,}` 은 앞선 패턴이 최소 `n`번 이상 반복되는 문자열을 의미한다.

```
const target = `A AA B BB Aa Bb AAA`;
cosnt regExp = /A{1,2}/g;

// A가 최소 1번 최대 2번 반복되는 문자열을 전역 검색
target.match(regExp); // ["A", "AA", "A", "AA", A"]
```

- `+` 는 앞선 패턴이 최소 한번 이상 반복되는 문자열을 의미한다. = `{1,}`
- `?` 는 앞선 패턴이 최대 한 번(0번 포함) 이상 반복되는 문자열을 의미한다. = `{0,1}`

---

## 31.5.4 OR 검색

- `|` 는 `or` 의 의미를 갖는다.

```
const target = `A AA B BB Aa Bb`;
cosnt regExp = /A|B/g;

// A또는 B를 전역 검색
target.match(regExp); // ["A", "A", "A", "B", "B", "B",
"A", "B"]
```

☐ `[ ]` 내의 문자는 or로 동작한다.

- 범위를 지정하려면 `[ ]`내에 `-` 를 사용한다.

```
const target = `AA BB Aa Bb`;
cosnt regExp = /[A-Za-z]+/g;

// A~Z, a~z 가 1번 이상 반복되는 문자열을 전역 검색
target.match(regExp); // ["A", "BB", "Aa", "Bb"]
```

- `\d`는 `숫자` 를 의미한다.
- `\D` 는 `\d` 와 반대로 동작한다. (즉, 숫자가 아닌 문자를 의미)
- `\w` 는 `알파벳, 숫자, 언더스코어` 를 의미한다.
- `\W` 는 `\w` 와 반대로 동작한다. (즉, 알파벳, 숫자, 언더스코어가 아닌 문자를 의미)

---

## 31.5.5 NOT 검색

`[...]` `내`의 `^` 은 `not` 의 의미를 갖는다.

### 31.5.6 시작 위치로 검색

`[...]` `밖`의 `^` 은 `문자열의 시작` 을 의미 한다.

### 31.5.7 마지막 위치로 검색

`$` 은 `문자열의 마지막` 을 의미 한다.

---

## 31.6 자주 사용하는 정규표현식

### 31.6.1 특정 단어로 시작하는지 검사

- 'http://' 또는 'https://'로 시작하는지 검사

```
const url= 'https://example.com';

// [...]밖의 ^는 문자열의 시작을 의미
// | 는 or
// ? 는 앞선 패턴('s')가 0~1번 반복
/^https?:\/\//.test(url); // true
```

### 31.6.2 특정 단어로 끝나는지 검사

- 'html'로 끝나는지 검사

```
const fileName= 'idnex.html';

// $는 문자열의 마지막을 의미
/html$/.test(fileName); // true
```

### 31.6.3 숫자로만 이루어진 문자열인지 검사

- 숫자로만 시작하고 끝나는지 검사

```
const target= '12345';

// [...]밖의 ^는 문자열의 시작
// $ 는 문자열의 마지막
```

```
// \d 는 숫자
// + 는 앞선 패턴이 최소 한 번 이상 반복되는 문자열을 의미
/^\d+$/.test(target); // true
```

### 31.6.4 하나 이상의 공백으로 시작하는지 검사

`\s` 는 여러 가지 공백문자를 의미한다.

```
const target= '  hi!';

/^[\s]+/.test(target); // true
```

### 31.6.5 아이디로 사용 가능한지 검사

- 알파벳 대소문자 또는 숫자로 시작하고 끝나며, 4~10자리인지 검사

```
const id= 'abc1234';

// [...]밖의 ^는 문자열의 시작
// $ 는 문자열의 마지막
// \d 는 숫자
// + 는 앞선 패턴이 최소 한 번 이상 반복되는 문자열을 의미
/^[A-Za-z0-9]{4,10}$/.test(id); // true
```